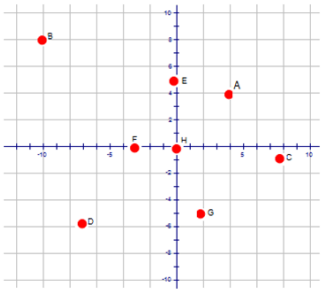
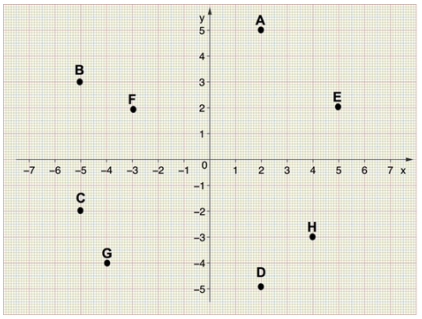
Fundamentación de robótica (Gpo 101) Sábado 6 de Mayo del 2023
José Jezarel Sánchez Mijares A01735226

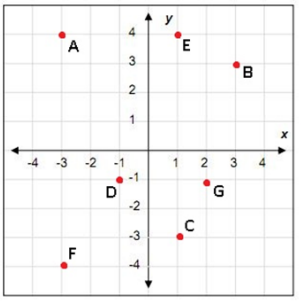
Actividad 5 (Landmarks)

Instrucciones:

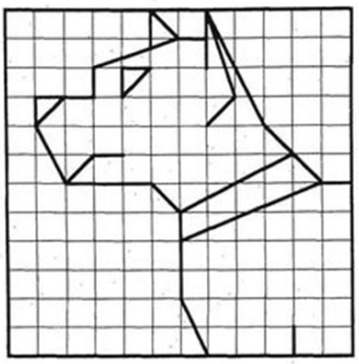
1. Implementar el código requerido para generar el seguimiento de los siguientes waypoints (puntos de referencia), ajustando el tiempo de muestreo: “sampleTime”, vector de tiempo: “tVec”, pose inicial: “initPose”, y los waypoints: “waypoints”

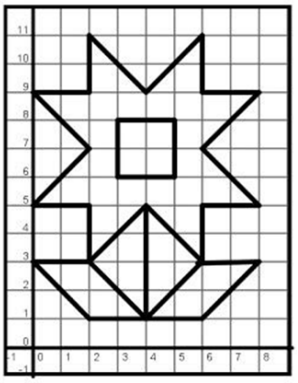


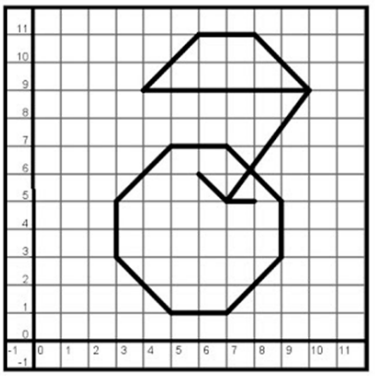




2. Generar los waypoints (puntos de referencia) necesarios para obtener las siguientes trayectorias, ajustando el tiempo de muestreo: “sampleTime”, vector de tiempo: “tVec”, pose inicial: “initPose”, y los waypoints: “waypoints”







Introducción:

La robótica inteligente se ocupa de diseñar y construir robots que puedan percibir, razonar y actuar de forma autónoma en entornos complejos y dinámicos. Un problema fundamental es el de la localización, que consiste en determinar la posición y orientación del robot con respecto a su entorno. Para ello, los robots utilizan sensores que les permiten observar su propio movimiento y el de su entorno, y técnicas probabilísticas que les permiten combinar la información de los sensores con un modelo del movimiento y un mapa del entorno. En esta tarea se desarrolló un código que permitiera trazar rutas a un robot e ir guiando con el uso de controladores y rutas definidas.

Desarrollo:

Para completar esta actividad se utilizó como base el código proporcionado por el profesor, para poder graficar los puntos se usó un método heurístico que permitio la calibración de los controladores, así como el tiempo de muestreo que necesitaban

Lo primero que hacemos es establecer las dimensiones de nuestro robot, lo siguiente es trazar una

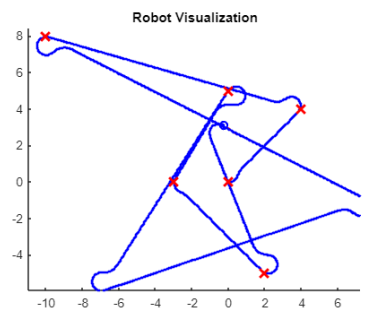
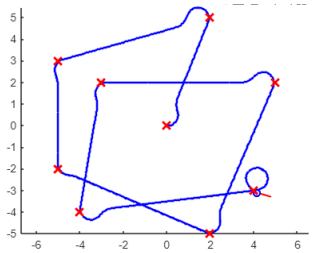
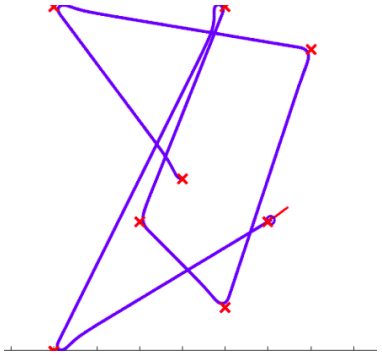
ruta, predefinida por nosotros, con el uso del algoritmo de Pure Pursuit.

Al ejecutar el programa este entra en un bucle de simulación que calcula la posición y velocidad a la que se desplaza el vehículo, vamos actualizando su posición dependiendo del sample time que le demos, así cuando se llega al punto ideal el robot actualiza su objetivo al siguiente objetivo

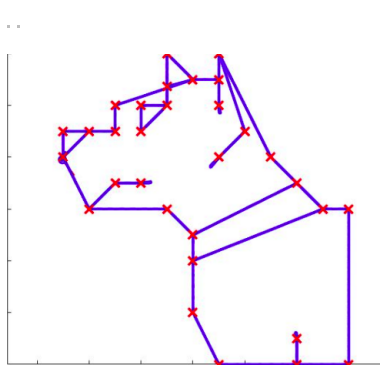
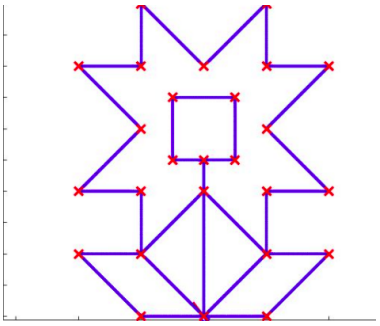
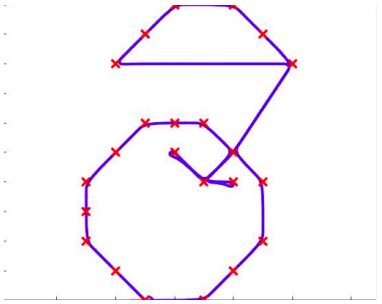
```
%% EXAMPLE: Differential drive vehicle following waypoints using the
% Pure Pursuit algorithm(Trayectoria 1)
%
% Copyright 2018-2019 The MathWorks, Inc.
%% Define Vehicle
R = 0.01; % Wheel radius [m]
L = 0.5; % Wheelbase [m]
dd = DifferentialDrive(R,L);
%% Simulation parameters
sampleTime = 0.1; % Sample time [s]
tVec = 0:sampleTime:130; % Time array
initPose = [0;0;0]; % Initial pose (x y theta)
pose = zeros(3,numel(tVec)); % Pose matrix
pose(:,1) = initPose;
% Define waypoints
waypoints = [0,0; 4,4; -10,8; 7.7,-1; -7,-6; 0,5; -3,0; 2,-5; 0,0];
% Create visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;
%% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.35;
controller.DesiredLinearVelocity = 0.75;
controller.MaxAngularVelocity = 1.5;
%% Simulation loop
close all
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)
% Run the Pure Pursuit controller and convert output to wheel speeds
[vRef,wRef] = controller(pose(:,idx-1));
[wL,wR] = inverseKinematics(dd,vRef,wRef);
% Compute the velocities
[v,w] = forwardKinematics(dd,wL,wR);
velB = [v;0;w]; % Body velocities [vx;vy;w]
vel = bodyToWorld(velB,pose(:,idx-1)); % Convert from body to world
% Perform forward discrete integration step
pose(:,idx) = pose(:,idx-1) + vel*sampleTime;
% Update visualization
viz(pose(:,idx),waypoints)
waitfor(r);
end
```

Resultados:

Parte 1

Ruta 1	Ruta 2	Ruta 3
		

Parte 2

Perro	Flor	Cereza
		

Cada una de las figuras obtenidas cuenta con un diferente número de puntos, así como distintas velocidades angulares y lineales, en algunos casos debido a la cantidad de puntos también tuvimos que mover la cantidad de puntos, sin embargo tenemos que tener en cuenta que no va a aplicar en todos igual por ejemplo la velocidad lineal debido a que si esta la aumentamos o disminuimos podremos ver un aumento en la ganancia del sobretiro en nuestro sistema.

Conclusión:

La robótica inteligente es un campo apasionante y desafiante que tiene muchas aplicaciones potenciales en diversos ámbitos de la sociedad. La localización es uno de los problemas más

importantes y difíciles que deben resolver los robots inteligentes para poder navegar y actuar de forma autónoma en entornos reales. En esta actividad tuve un mayor acercamiento a los problemas que esto implica además de conocer más sobre el funcionamiento de robots autónomos.

Referencias:

Kumar, A. (2020). Utilizing Mapping and Localization in SLAM for Robotics. Technical Articles. https://control.com/technical-articles/utilizing-mapping-and-localization-in-slam-for-robotics/